General estimation and evaluation of compositional distributional semantic models

Georgiana Dinu and **Nghia The Pham** and **Marco Baroni**
Center for Mind/Brain Sciences (University of Trento, Italy)
(georgiana.dinu|thenghia.pham|marco.baroni)@unitn.it

Abstract

In recent years, there has been widespread interest in *compositional* distributional semantic models (cDSMs), that derive meaning representations for phrases from their parts. We present an evaluation of alternative cDSMs under truly comparable conditions. In particular, we extend the idea of Baroni and Zamparelli (2010) and Guevara (2010) to use corpus-extracted examples of the target phrases for parameter estimation to the other models proposed in the literature, so that all models can be tested under the same training conditions. The linguistically motivated functional model of Baroni and Zamparelli (2010) and Coecke et al. (2010) emerges as the winner in all our tests.

1 Introduction

The need to assess similarity in meaning is central to many language technology applications, and distributional methods are the most robust approach to the task. These methods measure word similarity based on patterns of occurrence in large corpora, following the intuition that similar words occur in similar contexts. More precisely, vector space models, the most widely used distributional models, represent words as high-dimensional vectors, where the dimensions represent (functions of) context features, such as co-occurring context words. The relatedness of two words is assessed by comparing their vector representations.

The question of assessing meaning similarity *above the word level* within the distributional paradigm has received a lot of attention in recent years. A number of compositional frameworks have been proposed in the literature, each of these defining operations to combine word vectors into representations for phrases or even entire sentences. These range from simple but robust methods such as vector addition to more advanced methods, such as learning function words as tensors and composing constituents through inner product operations. Empirical evaluations in which alternative methods are tested in comparable settings are thus called for. This is complicated by the fact that the proposed compositional frameworks package together a number of choices that are conceptually distinct, but difficult to disentangle. Broadly, these concern (i) the input representations fed to composition; (ii) the composition operation proper; (iii) the method to estimate the parameters of the composition operation.

For example, Mitchell and Lapata in their classic 2010 study propose a set of composition operations (multiplicative, additive, etc.), but they also experiment with two different kinds of input representations (vectors recording co-occurrence with words vs. distributions over latent topics) and use supervised training via a grid search over parameter settings to estimate their models. Guevara (2010), to give just one further example, is not only proposing a different composition method with respect to Mitchell and Lapata, but he is also adopting different input vectors (word co-occurrences compressed via SVD) and an unsupervised estimation method based on minimizing the distance of composed vectors to their equivalents directly extracted from the source corpus.

Blacoe and Lapata (2012) have recently highlighted the importance of teasing apart the different aspects of a composition framework, presenting an evaluation in which different input vector representations are crossed with different composition methods. However, two out of three composition methods they evaluate are parameter-free, so that they can side-step the issue of fixing the parameter estimation method.

In this work, we evaluate *all* composition methods we know of, excluding a few that lag be-

hind the state of the art or are special cases of those we consider, *while keeping the estimation method constant*. This evaluation is made possible by our extension to all target composition models of the corpus-extracted phrase approximation method originally proposed in *ad-hoc* settings by Baroni and Zamparelli (2010) and Guevara (2010). For the models for which it is feasible, we compare the phrase approximation approach to supervised estimation with crossvalidation, and show that phrase approximation is competitive, thus confirming that we are not comparing models under poor training conditions. Our tests are conducted over three tasks that involve different syntactic constructions and evaluation setups. Finally, we consider a range of parameter settings for the input vector representations, to insure that our results are not too brittle or parameter-dependent.[1]

2 Composition frameworks

Distributional semantic models (DSMs) approximate word meanings with vectors recording their patterns of co-occurrence with corpus contexts (e.g., other words). There is an extensive literature on how to develop such models and on their evaluation (see, e.g., Clark (2012), Erk (2012), Turney and Pantel (2010)). We focus here on *compositional* DSMs (**cDSMs**). After discussing some options pertaining to the input vectors, we review all the composition operations we are aware of (excluding only the tensor-product-based models shown by Mitchell and Lapata (2010) to be much worse than simpler models),[2] and then methods to estimate their parameters.

Input vectors Different studies have assumed different distributional inputs to composition. These include bag-of-words co-occurrence vectors, possibly mapped to lower dimensionality with SVD or other techniques (Mitchell and Lapata (2010) and many others), vectors whose dimensions record the syntactic link between targets and collocates (Erk and Padó, 2008; Thater et al., 2010), and most recently vectors based on neural language models (Socher et al., 2011; Socher et al., 2012). Blacoe and Lapata (2012) compared the three representations on phrase similarity and paraphrase detection, concluding that "simple is best", that is, the bag-of-words approach performs at least as good or better than either syntax-based or neural representations across the board. Here, we take their message home and we focus on bag-of-words representations, exploring the impact of various parameters within this approach.

Model	Composition function	Parameters
Add	$w_1\vec{u} + w_2\vec{v}$	w_1, w_2
Mult	$\vec{u}^{w_1} \odot \vec{v}^{w_2}$	w_1, w_2
Dil	$\|\|\vec{u}\|\|_2^2\vec{v} + (\lambda - 1)\langle\vec{u}, \vec{v}\rangle\vec{u}$	λ
Fulladd	$W_1\vec{u} + W_2\vec{v}$	$W_1, W_2 \in \mathbf{R}^{m\times m}$
Lexfunc	$A_u\vec{v}$	$A_u \in \mathbf{R}^{m\times m}$
Fulllex	$tanh([W_1, W_2]\begin{bmatrix} A_u\vec{v} \\ A_v\vec{u} \end{bmatrix})$	$W_1, W_2,$ $A_u, A_v \in \mathbf{R}^{m\times m}$

Table 1: Composition functions of inputs (u, v).

Most frameworks assume that word vectors constitute rigid inputs fixed *before* composition, often using a separate word-similarity task independent of composition. The only exception is Socher et al. (2012), where the values in the input vectors are re-estimated during composition parameter optimization. Our re-implementation of their method assumes rigid input vectors instead.

Composition operations Mitchell and Lapata (2008; 2010) present a set of simple but effective models in which each component of the output vector is a function of the corresponding components of the inputs. Given input vectors $\vec{u}$ and $\vec{v}$, the weighted additive model (**Add**) returns their weighted sum: $\vec{p} = w_1\vec{u} + w_2\vec{v}$. In the dilation model (**Dil**), the output vector is obtained by decomposing one of the input vectors, say $\vec{v}$, into a vector parallel to $\vec{u}$ and an orthogonal vector, and then dilating only the parallel vector by a factor λ before re-combining (formula in Table 1). Mitchell and Lapata also propose a simple multiplicative model in which the output components are obtained by component-wise multiplication of the corresponding input components. We introduce here its natural *weighted* extension (**Mult**), that takes w_1 and w_2 powers of the components before multiplying, such that each phrase component p_i is given by: $p_i = u_i^{w_1}v_i^{w_2}$.

[1] We made the software we used to construct semantic models and estimate and test composition methods available online at `http://clic.cimec.unitn.it/composes/toolkit/`

[2] Erk and Padó (2008) and Thater et al. (2010) use input vectors that have been adapted to their phrasal contexts, but then apply straightforward composition operations such as addition and multiplication to these contextualized vectors. Their approaches are thus not alternative cDSMs, but special ways to construct the input vectors. Grefenstette and Sadrzadeh (2011a; 2011b) and Kartsaklis et al. (2012) propose estimation techniques for the tensors in the functional model of Coecke et al. (2010). Turney (2012) does not compose representations but similarity scores.

Guevara (2010) and Zanzotto et al. (2010) explore a full form of the additive model (**Fulladd**), where the two vectors entering a composition process are pre-multiplied by weight matrices before being added, so that each output component is a weighted sum of *all* input components: $\vec{p} = W_1\vec{u} + W_2\vec{v}$.

Baroni and Zamparelli (2010) and Coecke et al. (2010), taking inspiration from formal semantics, characterize composition as *function application*. For example, Baroni and Zamparelli model adjective-noun phrases by treating the adjective as a function from nouns onto (modified) nouns. Given that linear functions can be expressed by matrices and their application by matrix-by-vector multiplication, a functor (such as the adjective) is represented by a matrix A_u to be composed with the argument vector $\vec{v}$ (e.g., the noun) by multiplication, returning the *lexical function* (**Lexfunc**) representation of the phrase: $\vec{p} = A_u\vec{v}$.

The method proposed by Socher et al. (2012) (see Socher et al. (2011) for an earlier proposal from the same team) can be seen as a combination and non-linear extension of Fulladd and Lexfunc (that we thus call **Fulllex**) in which *both* phrase elements act as functors (matrices) *and* arguments (vectors). Given input terms *u* and *v* represented by $(\vec{u}, A_u)$ and $(\vec{v}, A_v)$, respectively, their composition vector is obtained by applying first a linear transformation and then the hyperbolic tangent function to the concatenation of the products $A_u\vec{v}$ and $A_v\vec{u}$ (see Table 1 for the equation). Socher and colleagues also present a way to construct matrix representations for specific phrases, needed to scale this composition method to larger constituents. We ignore it here since we focus on the two-word case.

Estimating composition parameters If we have manually labeled example data for a target task, we can use supervised machine learning to optimize parameters. Mitchell and Lapata (2008; 2010), since their models have just a few parameters to optimize, use a direct grid search for the parameter setting that performs best on the training data. Socher et al. (2012) train their models using multinomial softmax classifiers.

If our goal is to develop a cDSM optimized for a specific task, supervised methods are undoubtedly the most promising approach. However, every time we face a new task, parameters must be re-estimated from scratch, which goes against the idea of distributional semantics as a general similarity resource (Baroni and Lenci, 2010). Moreover, supervised methods are highly composition-model-dependent, and for models such as Fulladd and Lexfunc we are not aware of proposals about how to estimate them in a supervised manner.

Socher et al. (2011) propose an *autoencoding* strategy. Given a decomposition function that reconstructs the constituent vectors from a phrase vector (e.g., it re-generates *green* and *jacket* vectors from the composed *green jacket* vector), the composition parameters minimize the distance between the original and reconstructed input vectors. This method does not require hand-labeled training data, but it is restricted to cDSMs for which an appropriate decomposition function can be defined, and even in this case the learning problem might lack a closed-form solution.

Guevara (2010) and Baroni and Zamparelli (2010) optimize parameters using examples of how the output vectors should look like that are directly extracted from the corpus. To learn, say, a Lexfunc matrix representing the adjective *green*, we extract from the corpus example vectors of ⟨*N, green N*⟩ pairs that occur with sufficient frequency (⟨*car, green car*⟩, ⟨*jacket, green jacket*⟩, ⟨*politician, green politician*⟩, ...). We then use least-squares methods to find weights for the *green* matrix that minimize the distance between the *green N* vectors generated by the model given the input *N* and the corresponding corpus-observed phrase vectors. This is a very general approach, it does not require hand-labeled data, and it has the nice property that corpus-harvested phrase vectors provide direct evidence of the polysemous behaviour of functors (the *green jacket* vs. *politician* contexts, for example, will be very different). In the next section, we extend the **corpus-extracted phrase approximation** method to all cDSMs described above, with closed-form solutions for all but the Fulllex model, for which we propose a rapidly converging iterative estimation method.

3 Least-squares model estimation using corpus-extracted phrase vectors[3]

Notation Given two matrices $X, Y \in \mathbf{R}^{m\times n}$ we denote their inner product by $\langle X, Y\rangle$, ($\langle X, Y\rangle = \sum_{i=1}^{m}\sum_{j=1}^{n} x_{ij}y_{ij}$). Similarly we denote by $\langle u, v\rangle$ the dot product of two vectors $u, v \in \mathbf{R}^{m\times 1}$ and by $||u||$ the Euclidean norm of a vector:

[3] Proofs omitted due to space constraints.

$||u|| = \langle u, u\rangle^{1/2}$. We use the following Frobenius norm notation: $||X||_F = \langle X, X\rangle^{1/2}$. Vectors are assumed to be column vectors and we use x_i to stand for the i-th $(m \times 1)$-dimensional column of matrix X. We use $[X, Y] \in \mathbf{R}^{m\times 2n}$ to denote the horizontal concatenation of two matrices while $\begin{bmatrix} X \\ Y \end{bmatrix} \in \mathbf{R}^{2m\times n}$ is their vertical concatenation.

General problem statement We assume vocabularies of constituents $\mathcal{U}$, $\mathcal{V}$ and that of resulting phrases $\mathcal{P}$. The training data consist of a set of tuples (u, v, p) where p stands for the phrase associated to the constituents u and v:

$$T = \{(u_i, v_i, p_i)|(u_i, v_i, p_i) \in \mathcal{U}\times\mathcal{V}\times\mathcal{P}, 1 \leq i \leq k\}$$

We build the matrices $U, V, P \in \mathbf{R}^{m\times k}$ by concatenating the vectors associated to the training data elements as columns.[4]

Given the training data matrices, the general problem can be stated as:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} ||P - fcomp_{\boldsymbol{\theta}}(U, V)||_F$$

where $fcomp_{\boldsymbol{\theta}}$ is a composition function and $\boldsymbol{\theta}$ stands for a list of parameters that this composition function is associated to. The composition functions are defined: $fcomp_{\boldsymbol{\theta}} : \mathbf{R}^{m\times 1} \times \mathbf{R}^{m\times 1} \rightarrow \mathbf{R}^{m\times 1}$ and $fcomp_{\boldsymbol{\theta}}(U, V)$ stands for their natural extension when applied on the individual columns of the U and V matrices.

Add The weighted additive model returns the sum of the composing vectors which have been re-weighted by some scalars w_1 and w_2: $\vec{p} = w_1\vec{u} + w_2\vec{v}$. The problem becomes:

$$w_1^*, w_2^* = \arg\min_{w_1,w_2\in\mathbf{R}} ||P - w_1U - w_2V||_F$$

The optimal w_1 and w_2 are given by:

$$w_1^* = \frac{||V||_F^2\langle U, P\rangle - \langle U, V\rangle\langle V, P\rangle}{||U||_F^2||V||_F^2 - \langle U, V\rangle^2} \quad (1)$$

$$w_2^* = \frac{||U||_F^2\langle V, P\rangle - \langle U, V\rangle\langle U, P\rangle}{||U||_F^2||V||_F^2 - \langle U, V\rangle^2} \quad (2)$$

[4]In reality, not all composition models require u, v and p to have the same dimensionality.

Dil Given two vectors $\vec{u}$ and $\vec{v}$, the dilation model computes the phrase vector $\vec{p} = ||\vec{u}||^2\vec{v} + (\lambda - 1)\langle\vec{u}, \vec{v}\rangle\vec{u}$ where the parameter λ is a scalar. The problem becomes:

$$\lambda^* = \arg\min_{\lambda\in\mathbf{R}} ||P - VD_{||u_i||^2} - UD_{(\lambda-1)\langle u_i,v_i\rangle}||_F$$

where by $D_{||u_i||^2}$ and $D_{(\lambda-1)\langle u_i,v_i\rangle}$ we denote diagonal matrices with diagonal elements (i, i) given by $||u_i||^2$ and $(\lambda - 1)\langle u_i, v_i\rangle$ respectively. The solution is:

$$\lambda^* = 1 - \frac{\sum_{i=1}^k \langle u_i, (||u_i||^2 v_i - p_i)\rangle\langle u_i, v_i\rangle}{\sum_{i=1}^k \langle u_i, v_i\rangle^2||u_i||^2}$$

Mult Given two vectors $\vec{u}$ and $\vec{v}$, the weighted multiplicative model computes the phrase vector $\vec{p} = \vec{u}^{w_1} \odot \vec{v}^{w_2}$ where $\odot$ stands for component-wise multiplication. We assume for this model that $U, V, P \in \mathbf{R}_{++}^{m\times n}$, i.e. that the entries are strictly larger than 0: in practice we add a small smoothing constant to all elements to achieve this (Mult performs badly on negative entries, such as those produced by SVD). We use the w_1 and w_2 weights obtained when solving the much simpler related problem:[5]

$$w_1^*, w_2^* = \arg\min_{w_1,w_2\in\mathbf{R}} ||log(P) - log(U.^\wedge w_1 \odot V.^\wedge w_2)||_F$$

where $.^\wedge$ stands for the component-wise power operation. The solution is the same as that for Add, given in equations (1) and (2), with $U \rightarrow log(U)$, $V \rightarrow log(V)$ and $P \rightarrow log(P)$.

Fulladd The full additive model assumes the composition of two vectors to be $\vec{p} = W_1\vec{u} + W_2\vec{v}$ where $W_1, W_2 \in \mathbf{R}^{m\times m}$. The problem is:

$$[W_1, W_2]^* = \arg\min_{[W_1,W_2]\in\mathbf{R}^{m\times 2m}} ||P - [W_1W_2]\begin{bmatrix} U \\ V \end{bmatrix}||$$

This is a multivariate linear regression problem (Hastie et al., 2009) for which the least squares estimate is given by: $[W_1, W_2] = ((X^TX)^{-1}X^TY)^T$ where we use $X = [U^T, V^T]$ and $Y = P^T$.

Lexfunc The lexical function composition method learns a matrix representation for each functor (given by $\mathcal{U}$ here) and defines composition as matrix-vector multiplication. More precisely:

[5]In practice training Mult this way achieves similar or lower errors in comparison to Add.

$\vec{p} = A_u\vec{v}$ where A_u is a matrix associated to each functor $u \in \mathcal{U}$. We denote by T_u the training data subset associated to an element u, which contains only tuples which have u as first element. Learning the matrix representations amounts to solving the set of problems:

$$A_u = \underset{A_u \in \mathbf{R}^{m \times m}}{\arg\min} ||P_u - A_u V_u||$$

for each $u \in \mathcal{U}$ where $P_u, V_u \in \mathbf{R}^{m \times |T_u|}$ are the matrices corresponding to the T_u training subset. The solutions are given by: $A_u = ((V_u V_u^T)^{-1} V_u P_u^T)^T$. This composition function does not use the functor vectors.

Fulllex This model can be seen as a generalization of Lexfunc which makes no assumption on which of the constituents is a functor, so that both words get a matrix and a vector representation. The composition function is:

$$\vec{p} = tanh([W_1, W_2] \begin{bmatrix} A_u\vec{v} \\ A_v\vec{u} \end{bmatrix})$$

where A_u and A_v are the matrices associated to constituents u and v and $[W_1, W_2] \in \mathbf{R}^{m \times 2m}$. The estimation problem is given in Figure 1.

This is the only composition model which does not have a closed-form solution. We use a block coordinate descent method, in which we fix each of the matrix variables but one and solve the corresponding least-squares linear regression problem, for which we can use the closed-form solution. Fixing everything but $[W_1, W_2]$:

$$[W_1^*, W_2^*] = ((X^T X)^{-1} X^T Y)^T$$
$$X = \begin{bmatrix} [A_{u_1}\vec{v_1}, ..., A_{u_k}\vec{v_k}] \\ [A_{v_1}\vec{u_1}, ..., A_{v_k}\vec{u_k}] \end{bmatrix}^T$$
$$Y = atanh(P^T)$$

Fixing everything but A_u for some element u, the objective function becomes:

$$||atanh(P_u) - W_1 A_u V_u - W_2[A_{v_1}\vec{u}, ..., A_{v_{k'}}\vec{u}]||_F$$

where $v_1 ... v_{k'} \in \mathcal{V}$ are the elements occurring with u in the training data and V_u the matrix resulting from their concatenation. The update formula for the A_u matrices becomes:

$$A_u^* = W_1^{-1}((X^T X)^{-1} X^T Y)^T$$
$$X = V_u^T$$
$$Y = (atanh(P_u) - W_2[A_{v_1}\vec{u}, ..., A_{v_{k'}}\vec{u}])^T$$

In all our experiments, Fulllex estimation converges after very few passes though the matrices. Despite the very large number of parameters of this model, when evaluating on the test data we observe that using a higher dimensional space (such as 200 dimensions) still performs better than a lower dimensional one (e.g., 50 dimensions).

4 Evaluation setup and implementation

4.1 Datasets

We evaluate the composition methods on three phrase-based benchmarks that test the models on a variety of composition processes and similarity-based tasks.

Intransitive sentences The first dataset, introduced by Mitchell and Lapata (2008), focuses on simple sentences consisting of intransitive verbs and their noun subjects. It contains a total of 120 sentence pairs together with human similarity judgments on a 7-point scale. For example, *conflict erupts/conflict bursts* is scored 7, *skin glows/skin burns* is scored 1. On average, each pair is rated by 30 participants. Rather than evaluating against mean scores, we use each rating as a separate data point, as done by Mitchell and Lapata. We report Spearman correlations between human-assigned scores and model cosine scores.

Adjective-noun phrases Turney (2012) introduced a dataset including both noun-noun compounds and adjective-noun phrases (ANs). We focus on the latter, and we frame the task differently from Turney's original definition due to data sparsity issues.[6] In our version, the dataset contains 620 ANs, each paired with a single-noun paraphrase. Examples include: *archaeological site/dig*, *spousal relationship/marriage* and *dangerous undertaking/adventure*. We evaluate a model by computing the cosine of all 20K nouns in our semantic space with the target AN, and looking at the rank of the correct paraphrase in this list. The lower the rank, the better the model. We report median rank across the test items.

Determiner phrases The last dataset, introduced in Bernardi et al. (2013), focuses on a class of *grammatical* terms (rather than content

[6] Turney used a corpus of about 50 billion words, almost 20 times larger than ours, and we have very poor or no coverage of many original items, making the "multiple-choice" evaluation proposed by Turney meaningless in our case.

$$W_1^*, W_2^*, A_{u_1}^*, ..., A_{v_1}^*, ... = \arg\min_{\mathbf{R}^{m \times m}} ||atanh(P^T) - [W_1, W_2] \begin{bmatrix} [A_{u_1}\vec{v_1}, ..., A_{u_k}\vec{v_k}] \\ [A_{v_1}\vec{u_1}, ..., A_{v_k}\vec{u_k}] \end{bmatrix} ||_F$$
$$= \arg\min_{\mathbf{R}^{m \times m}} ||atanh(P^T) - W_1[A_{u_1}\vec{v_1}, ..., A_{u_k}\vec{v_k}] - W_2[A_{v_1}\vec{u_1}, ..., A_{v_k}\vec{u_k}]||_F$$

Figure 1: Fulllex estimation problem.

words), namely determiners. It is a multiple-choice test where target nouns (e.g., *amnesia)* must be matched with the most closely related determiner(-noun) phrases (DPs) (e.g., *no memory*). The task differs from the previous one also because here the targets are single words, and the related items are composite. There are 173 target nouns in total, each paired with one correct DP response, as well as 5 foils, namely the determiner (*no*) and noun (*memory*) from the correct response and three more DPs, two of which contain the same noun as the correct phrase (*less memory, all memory*), the third the same determiner (*no repertoire*). Other examples of targets/related-phrases are *polysemy/several senses* and *trilogy/three books*. The models compute cosines between target noun and responses and are scored based on their accuracy at ranking the correct phrase first.

4.2 Input vectors

We extracted distributional semantic vectors using as source corpus the concatenation of ukWaC, Wikipedia (2009 dump) and BNC, 2.8 billion tokens in total.[7] We use a bag-of-words approach and we count co-occurrences within sentences and with a limit of maximally 50 words surrounding the target word. By tuning on the MEN lexical relatedness dataset,[8] we decided to use the top 10K most frequent content lemmas as context features (vs. top 10K inflected forms), and we experimented with positive Pointwise and Local Mutual Information (Evert, 2005) as association measures (vs. raw counts, log transform and a probability ratio measure) and dimensionality reduction by Non-negative Matrix Factorization (NMF, Lee and Seung (2000)) and Singular Value Decomposition (SVD, Golub and Van Loan (1996)) (both outperforming full dimensionality vectors on MEN). For both reduction techniques, we varied the number of dimensions to be preserved from 50 to 300 in 50-unit intervals. As Local Mutual Information performed very poorly across composition experiments and other parameter choices, we dropped it. We will thus report, for each experiment and composition method, the distribution of the relevant performance measure across 12 input settings (NMF vs. SVD times 6 dimensionalities). However, since the Mult model, as expected, worked very poorly when the input vectors contained negative values, as is the case with SVD, for this model we report result distributions across the 6 NMF variations only.

4.3 Composition model estimation

Training by approximating the corpus-extracted phrase vectors requires corpus-based examples of input (constituent word) and output (phrase) vectors for the composition processes to be learned. In all cases, training examples are simply selected based on corpus frequency. For the first experiment, we have 42 distinct target verbs and a total of ≈20K training instances, that is, ⟨⟨*noun*, *verb*⟩, *noun-verb*⟩ tuples (505 per verb on average). For the second experiment, we have 479 adjectives and ≈1 million ⟨⟨*adjective*, *noun*⟩, *adjective-noun*⟩ training tuples (2K per adjective on average). In the third, 50 determiners and 50K ⟨⟨*determiner*, *noun*⟩, *determiner-noun*⟩ tuples (1K per determiner). For all models except Lexfunc and Fullex, training examples are pooled across target elements to learn a single set of parameters. The Lexfunc model takes only argument word vectors as inputs (the functors in the three datasets are verbs, adjectives and determiners, respectively). A separate weight matrix is learned for each functor, using the corresponding training data.[9] The Fulllex method jointly learns distinct matrix representations for both left- and right-hand side con-

[7] http://wacky.sslmit.unibo.it; http://www.natcorp.ox.ac.uk

[8] http://clic.cimec.unitn.it/~elia.bruni/MEN

[9] For the Lexfunc model we have experimented with least squeares regression with and without regularization, obtaining similar results.

stituents. For this reason, we must train this model on balanced datasets. More precisely, for the intransitive verb experiments, we use training data containing noun-verb phrases in which the verbs and the nouns are present in the lists of 1,500 most frequent verbs/nouns respectively, adding to these the verbs and nouns present in our dataset. We obtain 400K training tuples. We create the training data similarity for the other datasets obtaining 440K adjective-noun and 50K determiner phrase training tuples, respectively (we also experimented with Fulllex trained on the same tuples used for the other models, obtaining considerably worse results than those reported). Finally, for Dil we treat direction of stretching as a further parameter to be optimized, and find that for intransitives it is better to stretch verbs, in the other datasets nouns.

For the simple composition models for which parameters consist of one or two scalars, namely Add, Mult and Dil, we also tune the parameters through 5-fold crossvalidation on the datasets, directly optimizing the parameters on the target tasks. For Add and Mult, we search w_1, w_2 through the crossproduct of the interval $[0:5]$ in 0.2-sized steps. For Dil we use $\lambda \in [0:20]$, again in 0.2-sized steps.

5 Evaluation results

We begin with some remarks pertaining to the overall quality of and motivation for corpus-phrase-based estimation. In seven out of nine comparisons of this unsupervised technique with fully supervised crossvalidation (3 "simple" models –Add, Dil and Mult– times 3 test sets), there was no significant difference between the two estimation methods.[10] Supervised estimation outperformed the corpus-phrase-based method only for Dil on the intransitive sentence and AN benchmarks, but crossvalidated Dil was outperformed by at least one phrase-estimated simple model on both benchmarks.

The rightmost boxes in the panels of Figure 2 depict the performance distribution for using phrase vectors directly extracted from the corpus to tackle the various tasks. This non-compositional approach outperforms all compositional methods in two tasks over three, and it is one of the best approaches in the third, although in all cases even its top scores are far from the theoretical ceiling. Still, performance is impressive, especially in light of the fact that the non-compositional approach suffers of serious data-sparseness problems. Performance on the intransitive task is above state-of-the-art despite the fact that for almost half of the cases one test phrase is not in the corpus, resulting in 0 vectors and consequently 0 similarity pairs. The other benchmarks have better corpus-phrase coverage (nearly perfect AN coverage; for DPs, about 90% correct phrase responses are in the corpus), but many target phrases occur only rarely, leading to unreliable distributional vectors. We interpret these results as a good motivation for corpus-phrase-based estimation. On the one hand they show how good these vectors are, and thus that they are sensible targets of learning. On the other hand, they do not suffice, since natural language is infinitely productive and thus no corpus can provide full phrase coverage, justifying the whole compositional enterprise.

The other boxes in Figure 2 report the performance of the composition methods trained by corpus phrase approximation. Nearly all models are significantly above chance in all tasks, except for Fulladd on intransitive sentences. To put AN median ranks into perspective, consider that a median rank as high as 8,300 has near-0 probability to occur by chance. For DP accuracy, random guessing gets 0.17% accuracy.

Lexfunc emerges consistently as the best model. On intransitive constructions, it significantly outperforms all other models except Mult, but the difference approaches significance even with respect to the latter ($p = 0.071$). On this task, Lexfunc's *median* correlation (0.26) is nearly equivalent to the *best* correlation across a wide range of parameters reported by Erk and Padó (2008) (0.27). In the AN task, Lexfunc significantly outperforms Fullex and Dil and, visually, its distribution is slightly more skewed towards lower (better) ranks than any other model. In the DP task, Lexfunc significantly outperforms Add and Mult and, visually, most of its distribution lies above that of the other models. Most importantly, Lexfunc is the only model that is consistent across the three tasks, with all other models displaying instead a brittle performance pattern.[11]

Still, the top-performance range of all models

[10] Significance assessed through Tukey Honestly Significant Difference tests (Abdi and Williams, 2010), $\alpha = 0.05$.

[11] No systematic trend emerged pertaining to the input vector parameters (SVD vs. NMF and retained dimension number).

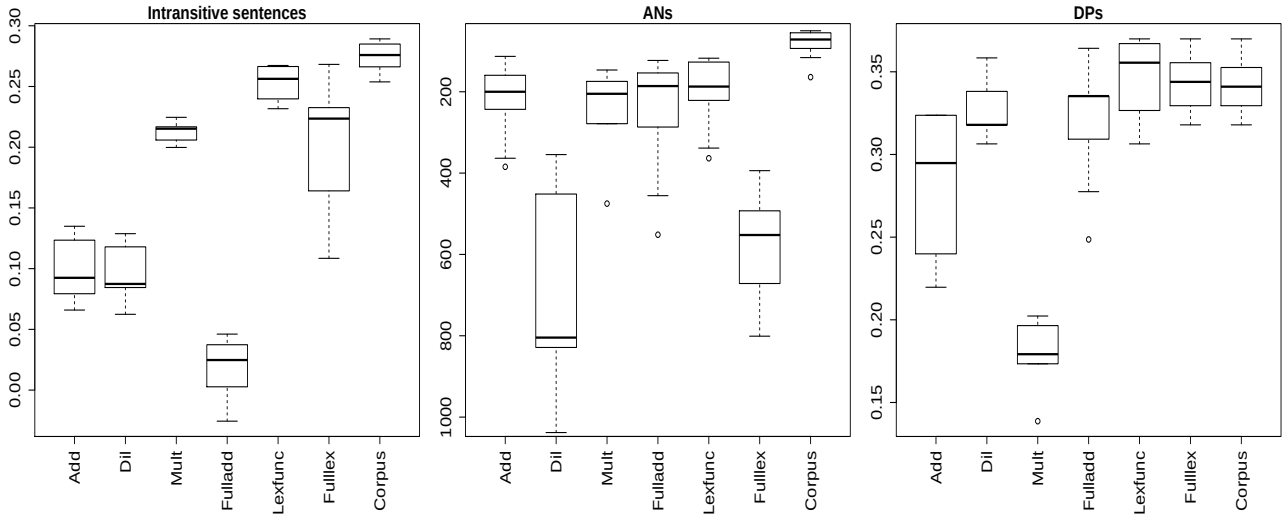

Figure 2: Boxplots displaying composition model performance distribution on three benchmarks, across input vector settings (6 datapoints for Mult, 12 for all other models). For intransitive sentences, figure of merit is Spearman correlation, for ANs median rank of correct paraphrase, and for DPs correct response accuracy. The boxplots display the distribution median as a thick horizontal line within a box extending from first to third quartile. Whiskers cover 1.5 of interquartile range in each direction from the box, and extreme outliers outside this extended range are plotted as circles.

on the three tasks is underwhelming, and none of them succeeds in exploiting compositionality to do significantly better than using whatever phrase vectors can be extracted from the corpus directly. Clearly, much work is still needed to develop truly successful cDSMs.

The AN results might look particularly worrying, considering that even the top (lowest) median ranks are above 100. A qualitative analysis, however, suggests that the actual performance is not as bad as the numerical scores suggest, since often the nearest neighbours of the ANs to be paraphrased are nouns that are as strongly related to the ANs as the gold standard response (although not necessarily proper paraphrases). For example, the gold response to *colorimetric analysis* is *colorimetry*, whereas the Lexfunc (NMF, 300 dimensions) nearest neighbour is *chromatography*; the gold response to *heavy particle* is *baryon*, whereas Lexfunc proposes *muon*; for *melodic phrase* the gold is *tune* and Lexfunc has *appoggiatura*; for *indoor garden*, the gold is *hothouse* but Lexfunc proposes *glasshouse* (followed by the more sophisticated *orangery*!), and so on and so forth.

6 Conclusion

We extended the unsupervised corpus-extracted phrase approximation method of Guevara (2010) and Baroni and Zamparelli (2010) to estimate all known state-of-the-art cDSMs, using closed-form solutions or simple iterative procedures in all cases. Equipped with a general estimation approach, we thoroughly evaluated the cDSMs in a comparable setting. The linguistically motivated Lexfunc model of Baroni and Zamparelli (2010) and Coecke et al. (2010) was the winner across three composition tasks, also outperforming the more complex Fulllex model, our re-implementation of Socher et al.'s (2012) composition method (of course, the composition method is only one aspect of Socher et al.'s architecture). All other composition methods behaved inconsistently.

In the near future, we want to focus on improving estimation itself. In particular, we want to explore ways to automatically select good phrase examples for training, beyond simple frequency thresholds. We tested composition methods on two-word phrase benchmarks. Another natural next step is to apply the composition rules recursively, to obtain representations of larger chunks, up to full sentences, coming, in this way, nearer to the ultimate goal of compositional distributional semantics.

Acknowledgments

We acknowledge ERC 2011 Starting Independent Research Grant n. 283554 (COMPOSES).

References

Hervé Abdi and Lynne Williams. 2010. Newman-Keuls and Tukey test. In Neil Salkind, Bruce Frey, and Dondald Dougherty, editors, *Encyclopedia of Research Design*, pages 897–904. Sage, Thousand Oaks, CA.

Marco Baroni and Alessandro Lenci. 2010. Distributional Memory: A general framework for corpus-based semantics. *Computational Linguistics*, 36(4):673–721.

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of EMNLP*, pages 1183–1193, Boston, MA.

Raffaella Bernardi, Georgiana Dinu, Marco Marelli, and Marco Baroni. 2013. A relatedness benchmark to test the role of determiners in compositional distributional semantics. In *Proceedings of ACL (Short Papers)*, Sofia, Bulgaria. In press.

William Blacoe and Mirella Lapata. 2012. A comparison of vector-based representations for semantic composition. In *Proceedings of EMNLP*, pages 546–556, Jeju Island, Korea.

Stephen Clark. 2012. Vector space models of lexical meaning. In Shalom Lappin and Chris Fox, editors, *Handbook of Contemporary Semantics, 2nd edition*. Blackwell, Malden, MA. In press.

Bob Coecke, Mehrnoosh Sadrzadeh, and Stephen Clark. 2010. Mathematical foundations for a compositional distributional model of meaning. *Linguistic Analysis*, 36:345–384.

Katrin Erk and Sebastian Padó. 2008. A structured vector space model for word meaning in context. In *Proceedings of EMNLP*, pages 897–906, Honolulu, HI.

Katrin Erk. 2012. Vector space models of word meaning and phrase meaning: A survey. *Language and Linguistics Compass*, 6(10):635–653.

Stefan Evert. 2005. *The Statistics of Word Cooccurrences*. Dissertation, Stuttgart University.

Gene Golub and Charles Van Loan. 1996. *Matrix Computations (3rd ed.)*. JHU Press, Baltimore, MD.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011a. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of EMNLP*, pages 1394–1404, Edinburgh, UK.

Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011b. Experimenting with transitive verbs in a DisCoCat. In *Proceedings of GEMS*, pages 62–66, Edinburgh, UK.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of GEMS*, pages 33–37, Uppsala, Sweden.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. 2009. *The Elements of Statistical Learning, 2nd ed.* Springer, New York.

Dimitri Kartsaklis, Mehrnoosh Sadrzadeh, and Stephen Pulman. 2012. A unified sentence space for categorical distributional-compositional semantics: Theory and experiments. In *Proceedings of COLING: Posters*, pages 549–558, Mumbai, India.

Daniel Lee and Sebastian Seung. 2000. Algorithms for Non-negative Matrix Factorization. In *Proceedings of NIPS*, pages 556–562.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL*, pages 236–244, Columbus, OH.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429.

Richard Socher, Eric Huang, Jeffrey Pennin, Andrew Ng, and Christopher Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*, pages 801–809, Granada, Spain.

Richard Socher, Brody Huval, Christopher Manning, and Andrew Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of EMNLP*, pages 1201–1211, Jeju Island, Korea.

Stefan Thater, Hagen Fürstenau, and Manfred Pinkal. 2010. Contextualizing semantic representations using syntactically enriched vector models. In *Proceedings of ACL*, pages 948–957, Uppsala, Sweden.

Peter Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188.

Peter Turney. 2012. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, 44:533–585.

Fabio Zanzotto, Ioannis Korkontzelos, Francesca Falucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of COLING*, pages 1263–1271, Beijing, China.